Building Global HEP Systems on Kerberos

Matt Crawford

Fermilab Computer Security

What this talk is…

- A variety of use cases for secure access by far-flung collaborations.
- An exploration of the security problems distributed systems must address.
- Examples of Kerberos-based solutions to those problems.

What this talk is not…

- Advocacy of one security mechanism over another.
- The final word on any of the topics that follow.

Quick Contrast of Kerberos and PK authentication

Kerberos	PKI
Principal holds secret key	End Entity holds private key
KDC issues tickets asserting secret key possession	CA issues certificates asserting public key binding
KDC knows all parties’ keys	CAs’ public keys known to all parties
TGTs reduce use of long-term client secret	Proxy certificates reduce use of long-term client secret
KDC must be on-line to client	Fresh CRLs or OCSP must be on-line to client & server

Problems to be Solved

- Web authentication
- Limited rights
- Unattended processes
- Shared agent authentication
- Long-queued and long-running jobs

Web Authentication

- Client host mounts /afs.
- User visits

  ```
  file:///afs/fnal.gov/files/expwww/…
  ```

- Browser knows nothing.
- Yes, it is a cheap trick.

Limited Rights

- Limited implementation of limited rights
  - Kernel support is typically poor-to-none
  - Storage systems are more flexible
- *user*/**afs**/*hostname@REALM* gets AFS the access of *user@REALM*.
- Kerberos tickets (& X.509 certificates) have room to invent something more.

Unattended Processes

- Unattended user processes (started by *cron*, for example) may need authenticated access.
- Using the user's own identity masks the dependency on host's integrity.
  - User does not have control of a stored secret key.
  - Keeping the user's own long-term key on-line is therefore not an option!
- How to manage this risk?
  - Make it explicit!

Expose the Risk

- Our solution:
  - *user@REALM* is authorized to create & destroy principals named *user*/**cron**/*host@REALM*
  - Keys are stored in private disk of *host*.
  - Initially these principals have no authorization, or have only AFS rights.
  - Can be added to ACL where needed.

Shared Agents

- Batch system or analysis farm initiates processes on behalf of many users.
- User processes may execute in many places.
- Users do not control (or know?) the security of their execution environment.
- User’s credentials could be compromised by an outsider or by another insider.
- Would like to be able to revoke and repair credentials put at risk.

Compute Farms

- Jobs on Fermilab farm *f* authenticate to services, claiming to act for user *u*, with principal *u*/*f*/**farm**@FNAL.GOV.
- Job submission is Kerberos-authenticated.
- Batch system obtains credentials for job.
- Farm principals are created by helpdesk, keys installed by support staff.

➔Does not scale !

Kerberized CAF System

- The CAF model is replicated ~25 times around the world.
- For each instance, security staff creates a special “headnode principal” which has the rights to create and destroy “CAF user principals.”
- As usual, CAF user principals have no rights except what users grant them.

Summary

- Kerberos is already widely used in HEP.
- It has been easy to build naming-based schemes to distinguish users and agents.
  - This allows management of risk in an environment of insecure systems, and a crude form of limited-rights authorization.
  - No protocol changes; some work on ACLs on the Kerberos administrative server.